9주차 예비보고서

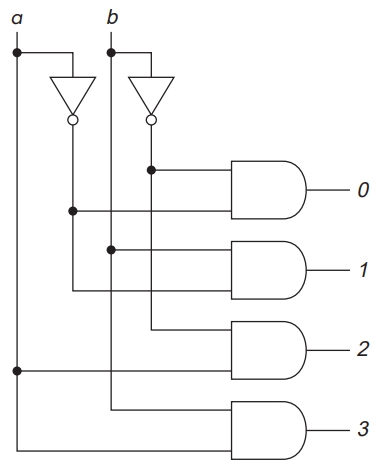
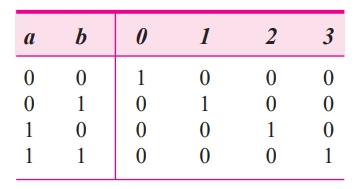
전공: 기계공학과 학년: 3학년 학번: 20191820 이름: 김형준

**1.**

Decoder는 활성화될 때 입력 신호에 따라 여러 개의 출력들 중 하나를 선택하는 논리회로이다. n비트의 input에 대해 개 이하의 output을 가지며, input의 값마다 단 하나의 output만 선택되어 활성화 된다.

선택된 output이 갖는 값에 따라 decoder의 종류가 나뉘는데, output이 1일 때 활성화 되는 decoder를 active high decoder라 하고, output이 0일 때 활성화 되는 decoder를 active low decoder라 한다.

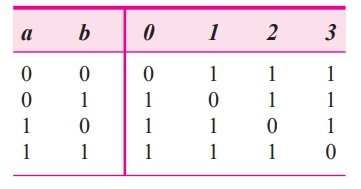
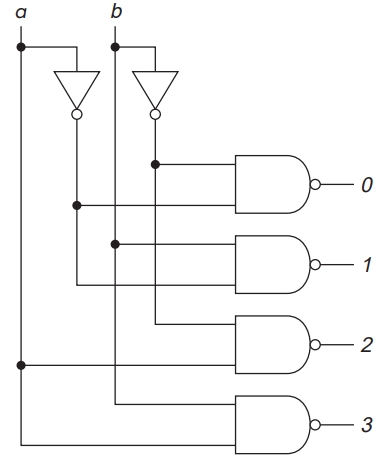
2 to 4 (2-input, 4-output) decoder를 예시로 들면, 아래 그림과 같이 NOT 게이트와 AND 게이트를 사용하여 active high decoder를 만들 수 있다.



[2 to 4 active high decoder의 회로도와 진리표]

위와 같이 입력 a, b의 값에 따라 출력 0, 1, 2, 3중 선택된 하나의 출력만 1로 활성화 되고, 나머지 출력들은 0으로 비활성화 된 것을 확인할 수 있다.

이때, 위의 회로도에서 AND 게이트를 모두 NAND 게이트로 바꾸면 아래 그림과 같이 2 to 4 active low decoder를 만들 수 있다. (active low decoder의 경우, 선택된 하나의 출력만 0으로 활성화 되고, 나머지 출력들은 1로 비활성화 됨)

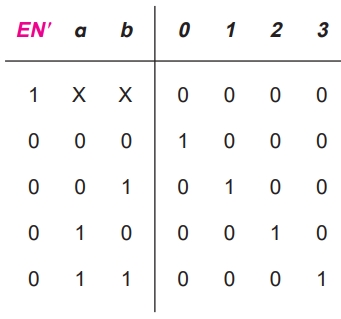
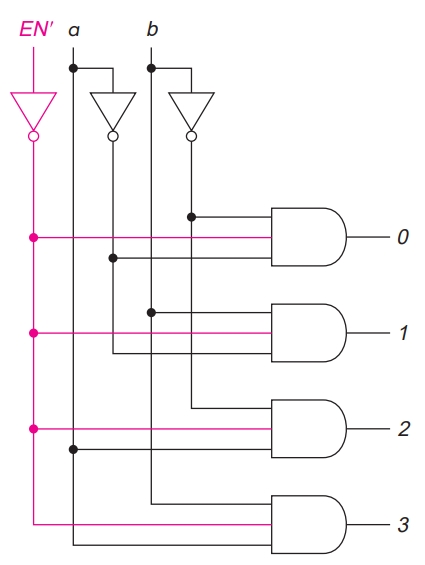


[2 to 4 active low decoder의 회로도와 진리표]

또한, 대부분의 decoder에는 하나 이상의 enable 입력을 가지고 있는데, 이 입력이 활성화 되면 그대로 작동하게 되고, 비활성화 되면 모든 출력이 비활성화 된다.

아래는 2 to 4 active high decoder에 active low enable 입력을 추가한 형태의 decoder의 진리표와 회로도이다.

enable 입력의 값이 1인 경우 input a, b의 값과 관계 없이 모든 output이 0으로 비활성화 되고, enable 입력의 값이 0인 경우 2 to 4 active high decoder와 동일하게 작동하는 것을 알 수 있다.



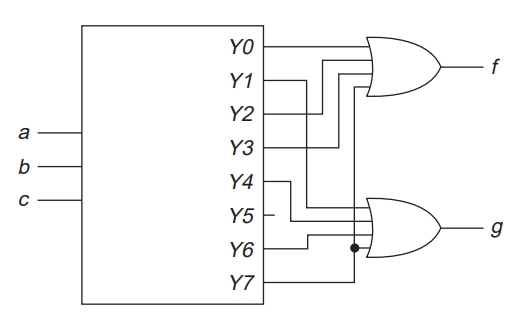
[active low enable input을 추가한 2 to 4 decoder의 회로도와 진리표]

한편, decoder는 특정 논리함수를 구현하는 용도로도 사용할 수 있는데, active high decoder의 각각의 output은 최소항(minterm)에 해당하므로 특정 논리함수를 SOP 형태로 변환한 다음, 필요한 minterm들을 OR 게이트를 사용해 연결하면 해당 논리함수를 구현할 수 있다.

아래와 같이 3 to 8 active high decoder와 OR 게이트를 사용해 ,

두 개의 논리함수를 각각 구현하면 아래와 같이 설계할 수 있다.

(Y0~Y7까지 각각 minterm 0~7번에 해당됨)

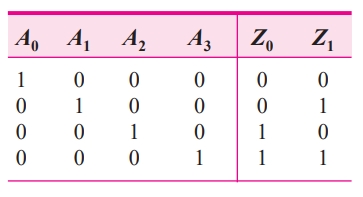
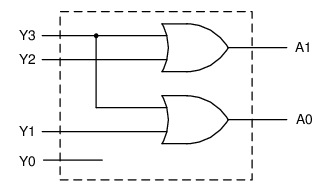


**2.**

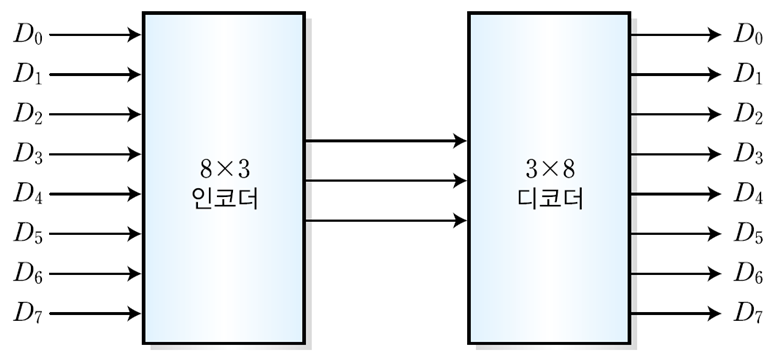
Encoder는 Decoder와 반대되는 동작을 하는 논리 회로로써, 개의 input을 받아 n개의 output으로 입력값에 대응하는 이진 코드를 출력한다.

Encoder의 Input은 Decoder의 output과 같고, Encoder의 output은 Decoder의 Input과 같다. 따라서 Decoder의 output처럼 Input들 중 1개만 값이 1인 입력만 valid하며, 나머지 입력값들은 모두 invalid input으로 처리한다. (invalid input인 경우 output은 don’t care이 됨)

예를 들어, 4 to 2 encoder의 진리표와 회로도는 아래와 같다.

****

decoder와 encoder의 관계를 회로도로 나타내면 다음과 같다.

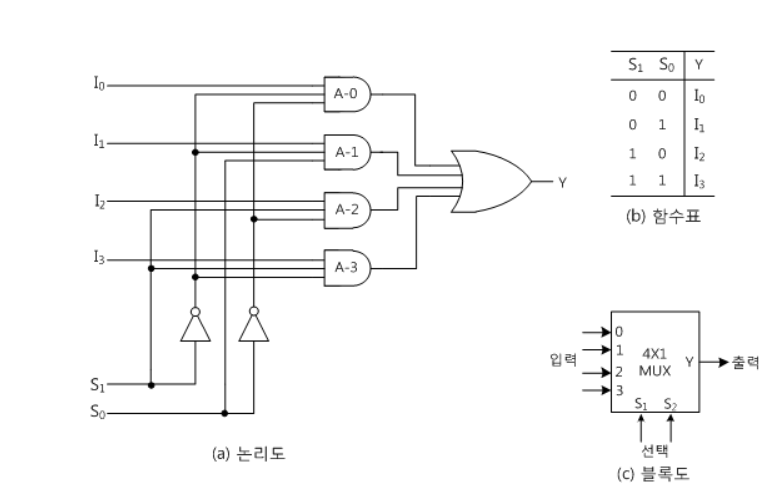


인코더를 사용해 암호화된 출력이 디코더의 입력으로 들어간다면, 원래의 입력값(인코더의 입력)이 디코더의 출력으로 그대로 나오므로 인코더로 암호화된 정보를 디코더를 사용해 원래대로 해독할 수 있다.

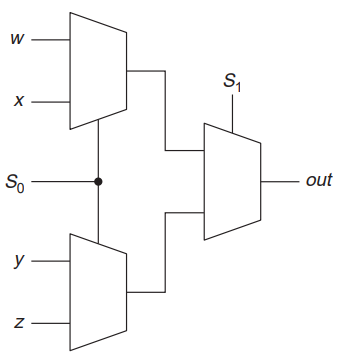
**3.**

Multiplexer는 줄여서 MUX라고 불리며, 다중 입력중의 하나를 선택 입력의 값에 따라 출력에 연결하는 스위치이다. 일반적인 경우, Multiplexer는 개의 input과 n개의 선택 입력으로 구성되어 있다. 이때, n개의 선택 입력들의 비트 조합에 따라 입력중 하나가 선택되어 output으로 출력된다.

아래는 4 to 1 Mux의 회로도, 진리표, 블록도이다.

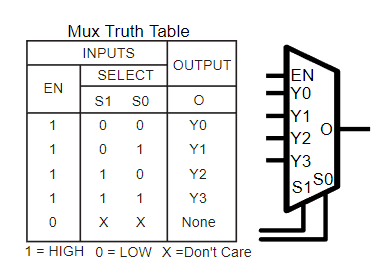


위 진리표와 같이 선택 입력 S0, S1에 의해 입력값 I0~I3중 하나가 선택되어 출력 Y의 값이 결정되는 것을 알 수 있다.

왼쪽 그림과 같이 위와 같은 4 to 1 Mux를 2 to 1 Mux 3개를 사용하여 구현할 수 있다.

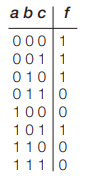
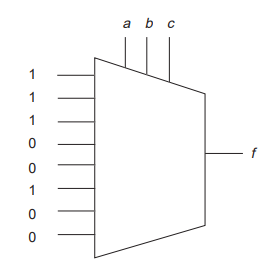
또한, decoder와 마찬가지로 Multiplexer에도 enable input을 가질 수 있는데, 이 경우 enable input이 비활성화 된 경우 출력은 항상 0이 되고, enable input이 활성화 되면 출력값은 enable input이 없는 Multiplexer와 동일하게 결정된다.

아래는 enable input이 있는 4 to 1 Mux의 진리표와 블록도이다.



Multiplexer는 논리 함수의 구현에도 사용될 수 있는데, 가장 단순한 논리함수 구현 방법은 선택 입력을 decoder처럼 사용하고, 데이터 입력에 0 또는 1을 연결하는 방식이다.

아래는 8 to 1 Mux를 사용하여 를 구현한 블록도와 진리표이다.



**4.**

하나 이상의 입력이 동시에 1이 되는 경우 오류를 방지하기 위해 입력간의 우선순위(Priority)를 추가해, 정해진 우선순위에 따라 출력을 결정하는 encoder를 priority encoder라고 한다.

Priority encoder에서 input에 1이 여러 개 인 경우, 가장 높은 우선순위를 가진 1만을 사용하여 output을 결정하고, 나머지 1들은 모두 무시된다. (don’t care로 처리)

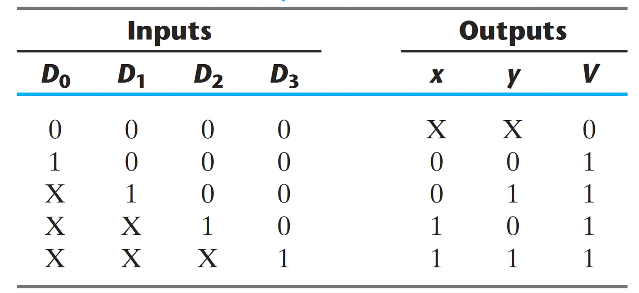
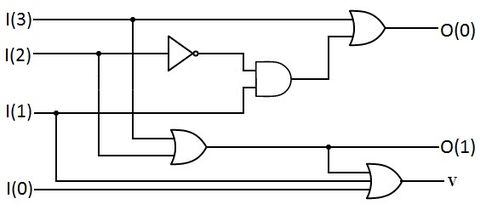
일반적으로 Priority encoder에서 우선순위는 내림차순 또는 오름차순으로 배정되며,

내림차순의 경우 가장 큰 입력 번호의 input이 가장 높은 우선순위를, 오름차순의 경우 가장 작은 입력 번호의 input이 가장 높은 우선순위를 가지게 된다.

출력값 중 V는 어떤 입력도 활성화 되지 않았다는 것을 표시하며, input이 모두 0인 경우 V의 값을 0으로 설정하고, 이외의 경우에는 1로 설정한다.

V의 값이 0인 경우, V를 제외한 나머지 출력값들은 신경쓰지 않는다. (don’t care로 처리)

아래는 priority encoder의 예시로, 오름차순으로 우선순위를 배정한 4 to 2 priority encoder의 회로도와 진리표이다.

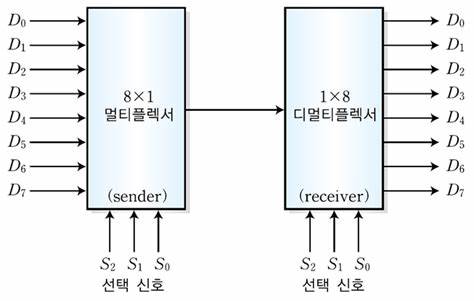


[4 to 2 priority encoder의 회로도와 진리표]

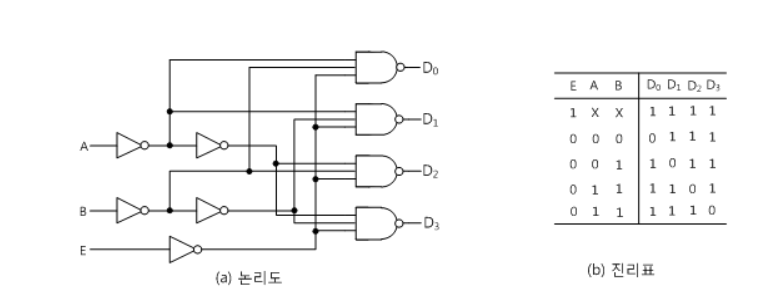
**5.**

DeMultiplexer(DeMux)는 Mux와 반대되는 동작을 수행하는 논리회로로, 하나의 input을 선택 입력의 값에 따라 개의 output 중 하나를 선택하여 출력한다.

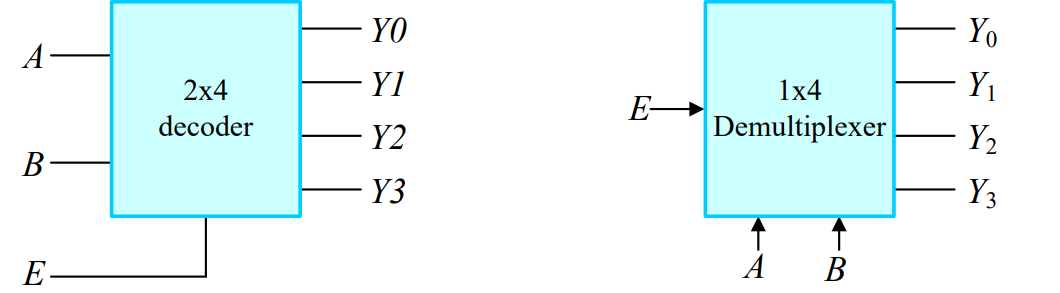
Multiplexer 와 DeMultiplexer 의 관계를 회로도로 나타내면 다음과 같다.



아래는 1 to 4 DeMux의 회로도와 진리표이다.



위의 회로도를 보면, 1 to 4 DeMux의 회로도와 2 to 4 Decoder의 회로도가 많이 유사하다는 것을 알 수 있다.



enable 입력이 있는 decoder는 DeMux로도 사용할 수 있는데, enable input이 있는 2 to 4 decoder에서 enable input을 DeMux의 input으로, 2개의 data input을 DeMux의 선택 입력으로 간주하면 1 to 4 DeMux로 작동하는 것을 알 수 있다. (enable 입력이 없는 decoder의 경우 DeMux로 사용할 수 없음)

이 경우, DeMux와 decoder는 개념적인 부분에서 구조와 사용처에 차이가 존재하게 된다.

Decoder는 n개의 입력에 의해서 최대 개의 출력을 결정한다. (input의 값마다 단 하나의 output만 선택되어 활성화 됨).

Decoder에 enable input이 있는 경우, enable이 활성화 될 때만 decoder로써 기능을 발휘한다. Decoder와 반대되는 기능을 하는 회로는 encoder이며, 주로 encoder에 의해 암호화된 정보를 원래대로 해독하는데 사용된다.

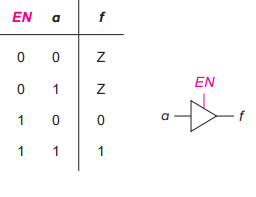
DeMux는 1개의 input을 n개의 선택 입력에 따라서 최대 개의 output 중 하나를 선택하여 연결한다. DeMux와 반대되는 기능을 하는 회로는 Multiplexer이며, 주로 입력된 데이터를 여러 개의 출력 중 원하는 곳으로 내보내는 데이터 분배와 스위칭에 사용된다.

**6.**

일반적인 논리 회로들의 입력과 출력값은 0 또는 1이었으나, 제어 입력(enable input)이 비활성화 되었을 때의 출력값을 Z로 표시하는 게이트를 3상(tristate) 게이트라 한다.

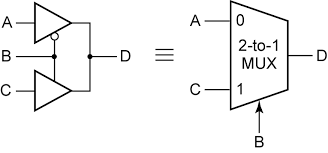
3상 게이트에서 enable input이 활성화 되었을 경우 게이트는 정상적으로 동작하지만, 비활성화 된 경우 게이트의 출력은 개회로(open circuit)처럼 연결되지 않은 것처럼 표현된다.

아래는 3상 버퍼 게이트의 진리표와 회로도이다.



위 진리표에서 EN(enable input)이 1(활성)이면 일반적인 buffer 게이트처럼 작동하나, EN이 0(비활성)이면 출력값이 입력(a)과 관계없이 항상 Z(high Impedance)로 표기되는 것을 확인할 수 있다.

위와 같은 3상 버퍼 게이트를 아래와 같이 조합하여 2 to 1 Mux를 구성할 수 있다.



위 그림에서 B(enable input)의 값이 0이면 A가 출력되고, B의 값이 1이면 C가 출력되기 때문에 위 그림처럼 3상 버퍼 게이트 2개를 사용하여 만든 조합회로는 2 to 1 Mux와 동일한 기능을 한다고 볼 수 있다.